1

2

3   **UNITED STATES PATENT APPLICATION**

4   **FOR**

5

6   **SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR PROVIDING A**

7   **REMOTE SUPPORT SERVICE**

8

9

10

11   Inventors:

12

13   Friedemann ULMER

14   Manfred LANGE

15   Thomas TRENZ

1

FIELD OF THE INVENTION

3

4   The present invention relates generally to information technological (IT)
5 infrastructure support, and more particularly to a system and a method for providing a
6 remote support service between at least one support-service provider's site and a
7 customer's site having a customer's IT infrastructure. The invention relates also to a
8 computer system forming a customer based part of a system for providing a remote
9 support service and a corresponding computer program product.

10

BACKGROUND OF THE INVENTION

12

13   For most companies the network (Internet or intranet) is the most critical and
14 often the most complicated element of their entire IT infrastructure. Proprietary or
15 customized networks therefore have to be maintained by way of support services in
16 order to maximize return on investment. These support services are delivered by
17 technical specialists, either locally or remotely.

18   For support services to a local area network (LAN), there are network
19 management systems available which provide detailed LAN health checks utilizing
20 passive monitoring probes. Full analysis and any errors or capacity problems
21 identified are documented in a report. They further provide remote LAN monitoring by
22 a remote log-in via a network or an ISDN connection. Monthly reports can detail
23 utilization errors and protocol use. A recommendation on solutions to any problems
24 identified may be included in the report. In addition, the service can include alarm
25 recording and problem diagnosis.

26   The present applicant offers an enterprise-oriented Network Node Manager
27 (OpenView™) which has a Web interface. In particular, a Web browser is shipped
28 together with the OpenView™ Professional Suite, which is a comprehensive software
29 solution that allows customers in small to midsize networked environments to
30 manage virtually all elements of a LAN. Thus the OpenView™ Professional Suite

1 combines the power of a central management console with the ease of using the

2 Web for communication.

3 Network management of a TCP/IP network comprises network management

4 stations (managers) communicating with network elements. The network elements

5 can be anything that runs the TCP/IP protocol suite: hosts, routers, terminal servers,

6 etc. (Regarding the meaning of the term "TCP/IP protocol suite", see W. Richard

7 Stevens: TCP/IP Illustrated, Volume 1, The Protocols, 1994, pages 1-2). The protocol

8 for the communication between the manager and the elements provided by the

9 TCP/IP protocol suite is the Simple Network Management Protocol (SNMP). It allows

10 a two-way communication: a manager can ask an element for a specific value, or the

11 element can tell the manager that something happened. Also, the manager is able to

12 set variables in the element, in addition to reading variables from it. A description of

13 SNMP can, for example, be found in Stevens, pages 359-388.

14 Another standard for network management is what is called Desktop

15 Management Interface (DMI). It has been defined by the Distributed Management

16 Task Force (DMTF). DMI is a standard framework for managing and tracking

17 components in a desktop personal computer, notebook or server (see

18 http://www.dmtf. org/spec/dmis.html).

19 An emerging standard for the management of operating systems and

20 applications is the Web-Based Enterprise Management (WBEM). WBEM is a set of

21 management tools using emerging technologies such as CIM and XML. In particular,

22 WBEM is a set of the following technologies: "CIM Schema v 2.2", "CIM operations

23 over HTTP", and "XML encodings for CIM" (see

24 http://www.dmtf.org/wbem/index.html). CIM stands for Common Information Model

25 and is a data model for describing information for the management of enterprise

26 computing environments. XML stands for Extensible Markup Language and is a

27 standard which can be used for exchanging massages between different applications

28 (see http://www.w3.org/tr/rec-xml).

29

30 SUMMARY OF THE INVENTION

31

1   A system for providing a remote support service between at least one support-
2   service provider's site and a customer's site having a customer's IT infrastructure,
3   comprises: an information collecting component which collects information about the
4   customer's IT infrastructure; a storage component which stores collected information
5   according to a data model modeling at least part of the customer's IT infrastructure;
6   an information-transferring component capable of transferring at least part of the
7   collected or stored information or a representation of it to the support-service
8   provider; and an analysis component which analyzes the stored or transferred
9   information or representation as a basis for the provision of the remote support
10   services.

11   According to another aspect, the invention provides a computer system forming
12   a customer based part of a system for providing a remote support service between at
13   least one support-service provider's site and the customer's site having a customer's
14   IT infrastructure. The computer system comprises: an information collecting
15   component which collects information about the customer's IT infrastructure; a
16   storage component which stores collected information according to a data model
17   modeling at least part of the customer's IT infrastructure; and an information-
18   transferring component capable of transferring at least part of the collected or stored
19   information or of a consolidated representation of it to the support-service provider.

20   According to still another aspect, the invention is directed to a computer
21   program product including program code for execution on a customer-based
22   computer system which is part of a system for providing a remote support service
23   between at least one support-service provider's site and the customer's site having a
24   customer's IT infrastructure. The program code comprises the software parts of:
25   an information collecting component which collects information about the
26   customer's IT infrastructure; a storage component which stores collected
27   information according to a data model modeling at least part of the customer's
28   IT infrastructure; an information-transferring component capable of transferring
29   at least part of the collected or stored information or of a consolidated
30   representation of it to the support-service provider.

31   According to yet another aspect, the invention is directed to a method for

1   providing a remote support service between at least one support-service provider's

2   site and a customer's site having a customer's IT infrastructure. The method

3   comprises the steps of: collecting information about the customer's IT infrastructure;

4   storing collected information according to a data model modeling at least part of the

5   customer's IT infrastructure; transferring at least part of the collected or stored

6   information or a representation of it to the support-service provider; and analyzing

7   the stored or transferred information or representation as a basis for the provision of

8   the remote support services.

9   Other features are inherent in the disclosed system, computer program

10 product and method or will become apparent to those skilled in the art from the

11 following detailed description of embodiments and its accompanying drawings.

12

13   DESCRIPTION OF THE DRAWINGS

14

15 In the accompanying drawings:

16   Fig. 1   shows an architectural overview of a preferred embodiment;

17   Fig. 2   shows an architectural representation of parts of an infrastructure

18 documentation tool of Fig. 1;

19   Fig. 3   shows a more detailed functional architecture of a data collector ;

20   Fig. 4   shows a more detailed functional architecture of a collection

21 configuration component;

22   Fig. 5   shows a more detailed functional architecture of a transport office

23 manager;

24   Fig. 6   illustrates a distributed application stack of a customer's IT

25 infrastructure;

26   Fig. 7   shows a graphical representation of an instance of a data model

27 modeling a customer's IT infrastructure;

28   Fig. 8   illustrates a data model of a network node;

29   Fig. 9   illustrates a data model of a computer system;

30   Fig. 10   illustrates a data model of a storage system;

31   Fig. 11   illustrates the inheritance of the data models of Figs. 8 to 10 from

1   more general classes;

2   Fig. 12   illustrates how an element of the IT infrastructure is mapped to a

3   class with dynamic attributes;

4   Fig. 13   is a table illustrating the concept of meta classes for achieving

5   dynamic extensibility of the data model;;

6   Fig. 14   shows a flow diagram of a remote support-service method;

7   Fig. 15   illustrates an embodiment wherein the support service is provided by

8   several co-operating sub-services;

9   Fig. 16   illustrates a collection task;

10   Fig. 17   illustrates a scheduling task;

11   Fig. 18   shows collectible interfaces;

12   Fig. 19a-c show further interfaces; and

13   Fig. 20   illustrates a DMI collection task.

14

15   DESCRIPTION OF THE PREFERRED EMBODIMENTS

16

17   Features that are substantially or functionally equal or similar will be referred to

18   with the same reference sign(s).

19   Figure 1 shows an architectural overview of a preferred embodiment. Before

20   proceeding further with the description, however, a few items of the preferred

21   embodiments will be discussed.

22   The preferred embodiments of the system allow for an automatic capture of

23   configuration and performance information of the customer's IT infrastructure via a

24   data collection mechanism which is independent of hardware devices. The process

25   runs as a background task. The collected information is stored in a storage

26   component according to a data model which models at least part of the entire IT

27   infrastructure which includes, but is not limited to network interconnect hardware and

28   related software. Preferably, the data model models the whole infrastructure. Based

29   on that data model, an analysis component, located at the service provider's site

30   and/or the customer's site analyzes the collected or stored information or a

31   representation of it.

1   In the preferred embodiments, a storage component is located at the customer's

2   site since access to the customer's site from outside is normally restricted or

3   excluded due to security requirements. In addition, there is another storage

4   component located at the provider's site. In other embodiments, the storage

5   component storing collected data according to the data model is only located at one

6   of the sites, either the support-service provider's site or the customer's site.

7   In the preferred embodiments, the analysis component is located at the support-

8   service provider's site; i.e. the stored data or an extract of them are transmitted to the

9   support-service provider's site and are analyzed there. The analysis can be

10   individually tailored to the customer, depending on the particular support contract

11   between customer and provider. The support-service provider receives data from the

12   customer, preferably via the Internet using e-mail, HTTP or a point-to-point Internet

13   connection, performs the diagnosis and sends a report or message back to the

14   customer, again via the Internet for example by sending XML. However, it is likewise

15   possible that the analysis component is located at the customer's site and the

16   analysis is being done there. Then, the results of the analysis are transferred to the

17   service provider's site, where the support-service server can, for example,

18   automatically arrange for service personnel to be sent to the customer, if the result

19   indicates a fault condition. Further, it is possible that a customer is linked to several

20   co-operating support-service providers and transfers data to them. For example,

21   each provider could be responsible only for certain IT infrastructure elements (for

22   example, for certain hardware devices). There could also be a hierarchical structure

23   of support-service providers in the sense that there are several sub-providers

24   (responsible only for providing support for certain parts of the IT infrastructure) and

25   one master provider (responsible for providing an overall support).

26   If the bandwidth of the network link(s) between the customer and the service

27   provider(s) is limited, it may be advantageous to consolidate the data before they are

28   transferred to the provider. If bandwidth limitations are not relevant, a consolidation

29   can likewise be performed at the provider. It is also possible to perform consolidation

30   actions at both sites. Consolidating means compressing data, e.g. by filtering or

31   condensing them or by detecting certain events.

1   The customer's IT infrastructure that can be discovered, monitored and
2   analyzed by the disclosed embodiments is not restricted to hardware. Rather, it may
3   comprise one or more of the following elements: network infrastructure elements,
4   storage systems, hardware elements and peripherals, operating systems, networking
5   software, database software, middleware and utilities, software applications. The
6   information collecting component collects information about at least one of these
7   elements and the data model models at least part of these elements and their inter-
8   relations.

9   The preferred embodiments of the system further comprise a discovery
10   component which is capable of automatically discovering changes in the customer's
11   IT infrastructure. There are many sources of ongoing changes in an IT infrastructure,
12   for example: Failure of infrastructure elements, fixing of failed infrastructure
13   elements, extensions and enhancements of the infrastructure, user process changes,
14   application changes, interface changes, installation or activation of new applications
15   and software modules, version upgrades, inclusion of new organization units, etc.
16   The data model is automatically adapted so that it models the changed IT
17   infrastructure. Owing to the automatic discovering capability of the discovery
18   component, after an installation of a program code representing the software part
19   of the system at the customer's site, the system can automatically discover at least
20   part of the customer's IT infrastructure and automatically and dynamically generate
21   and stores data which represent it.

22   In order to allow this dynamic generation and modification in the preferred
23   embodiments, the elements of the customer's IT infrastructure are mapped to classes
24   of an object-oriented programming language (i.e., they become instances of those
25   classes), and new classes (instances) can dynamically be added. The classes have
26   flexible attributes which can dynamically be added and changed during the execution
27   of the program. This is advantageous for the system's capability to automatically
28   adapt itself to changes in the IT infrastructure.

29   In a preferred embodiment, the information-transferring component comprises
30   transport managing means whereby the collected configuration information is
31   transferred via an information network, particularly the Internet, or by means of a data

1 carrier. An IT infrastructure support service can therefore be handled as an electronic
2 service as part of electronic commerce and business. The proposed web-based
3 approach facilitates the provision compatibility, platform-independence and high
4 accessibility.

5 As already mentioned above, a scaleable storage component, in particular an
6 object-oriented database is provided. Using a scaleable database allows an unlimited
7 growth of the IT infrastructure.

8 The storage component may be capable of storing performance history
9 information for the IT infrastructure. This facilitates the monitoring and/or analyzing of
10 the IT infrastructure and the assessment whether the infrastructure performance can
11 be enhanced through updates of the infrastructure hardware and software. Further,
12 history information allows improved diagnosis and performance checks. The
13 configuration, configuration changes, performance and/or performance changes of
14 the customer's IT infrastructure are automatically monitored and analyzed particularly
15 based on rules. Such rules define what checks and configuration test are to be
16 performed are to be performed in an infrastructure element of a particular type. The
17 rule are not "hard coded". Rather they can be input as ASCII strings and are
18 interpreted (similar to a script language such as VisualBasic Script). Additionally it is
19 possible with the preferred embodiments to monitor infrastructure health, including
20 but not limited to, trend analysis, forecasts, traffic assessment and problem
21 prediction.

22 In the preferred embodiments, a scheduler for scheduling the collection of the
23 infrastructure information is provided. The scheduler determines when a collection
24 task is be carried out.

25 The preferred embodiments of the computer program product comprise
26 program code which, for example, is stored on a computer-readable data carrier
27 or is in the form of signals transmitted over a computer network. The preferred
28 embodiments of the program code are written in an object-oriented programming
29 language (Java or C++). Some of the mentioned components have also a
30 hardware part. For example, the storage component comprises a physical
31 storage medium for persistently storing the collected data. It is clear that the

1   computer program product comprises only the software parts of these

2   components.

3   Returning now to Fig. 1, it shows an architectural overview of preferred

4   embodiment of a computer system for providing a remote support service. The

5   system is subdivided into an Infrastructure Documentation Tool (IDT) 1 at the

6   customer's site and an Infrastructure Support Service Tool (ISST) 2 on the support-

7   service provider's site. The two sites are linked via a network, for example an IP link

8   using HTTP 3, e-mail 4 or a point-to-point Internet connection (not illustrated).

9   The customer's IT infrastructure generally comprises network infrastructure

10   elements (such as routers and switches), storage systems, hardware elements (such

11   as desk top computers), peripherals (such as printers and scanners). Further, it

12   generally comprises software elements, such as operating systems, networking

13   software, database software, middleware, utilities and software applications. In Fig.

14   1, the customer's IT infrastructure 5 is visualized as a tree-like structure, but, more

15   generally, it can be a graph-like structure.

16   The several functional elements of the IDT 1 are controlled by an IDT controller

17   6,. which is the heart of the IDT 1. It controls a discovery component 7 which runs

18   automatically and periodically as a background task, for example once per day (a

19   component with such a function is also called a "demon"). The discovery component

20   is capable of discovering an appearance, disappearance or a change in

21   infrastructure elements, such as routers, switches, hosts, software applications etc.

22   The discovery component 7 sends requests to (yet unknown) infrastructure elements

23   by using what is called the Ping application (see Stevens, pages 85 to 96). In order

24   to discover unknown infrastructure elements it sends many trial requests with

25   different possible IP addresses, which possibly do not exist in the infrastructure 5. If,

26   by chance, it uses the IP address of the (yet unknown) infrastructure element, this

27   element will respond and disclose its identity in the response. The discovery

28   component 7 uses a network management protocol, such as SNMP, to discover

29   network elements, such as routers. In order to discover software elements, it uses a

30   suitable protocol, such as WBEM.

31   Ping may not be the optimal solution, if the subnet contains many devices, e.g.

1 when the subnet mask is big, e.g. 255.255.0.0. In this case reading the ARP cache
2 (see Stevens, pages 53 to 64, in particular page 56) is preferred. An ARP cache
3 contains the resolved network addresses of other devices with which a recent
4 communication across the network took place. Best candidates for reading ARP
5 caches are therefore gateways or servers.

6 The discovery components 7 starts the discovery task "within itself", i. e. on the
7 IDT server computer on which it runs, and first discovers the gateway to the
8 infrastructure 5. Then, it discovers the first node in the infrastructure 5 and receives
9 the requested identity information from it, by means of the above-described method.
10 Then, this node constitutes the starting point for the discovery of the adjacent nodes.
11 If the node is a switch, it provides the discovery component 7 with information as to
12 which device is linked to which port of the switch. If the infrastructure element is a
13 router, the above-described discovery method can be simplified as routers commonly
14 store a list of used IP addresses in their cache. If the discovery component 7 can
15 acquire such a stored list of used IP addresses, it can use them for the further
16 discovery task rather than trying a large number of arbitrary IP addresses.

17 By this method, the discovery component 7 not only discovers the elements of
18 the infrastructure 5, but also their inter-relations, which define the topology of the
19 infrastructure 5. In Fig. 1 the topology is illustrated as a tree-like structure.

20 The discovered infrastructure elements and the infrastructure topology are
21 mapped to objects of an object-oriented data model. The objects are persistently
22 stored in an object-oriented database, which forms part of a storage component 8.
23 The schema of the data base (i.e. the data model) can be dynamically modified.
24 Thus, if the discovery component 7 discovers a modification of the infrastructure 5
25 (e.g. the appearance, disappearance or a change of attributes of an infrastructure
26 element), it is not necessary to create a new database schema. Rather, the database
27 schema already used by the storage component 8 is modified correspondingly, e.g.
28 by dynamically adding or removing an object or changing an object's attributes list.

29 A data collector 9 collects information about the infrastructure 5 on the basis of
30 the discovered elements and the discovered infrastructure topology. The collected
31 information is mainly configuration and performance information. The collected data

1   are stored in the storage component 8 according to the data model.

2   An information-transferring component 10a, here denoted as transport office
3   manager, transfers data collected by the data collector 9 and stored in the storage
4   component 8 to the ISST 2 via the HTTP link 3 or the e-mail link 4.

5   The data collector 9 is configured by a collection configuration component 11. A
6   core service component 12 allows the configuration and debugging of the IDT 1. A
7   web server 13 permits an HTTP access to the IDT 2, for example by an infrastructure
8   administrator or configurator or the support-service provider.

9   The ISST 2 at the support-service provider's site comprises a transport office
10   manager 10b which is the counterpart of the IDT's transport office manager 10a. It
11   receives the infrastructure collection and topology data sent by the transport office
12   manager 10a. These data are stored in an ISST storage component 14 via an import
13   service component 15. Also in the ISST storage component 14, the data are stored
14   according to the data model.

15   An analysis component 16 analyses the topology and collected data with regard
16   to the particular support service to be provided to the customer. For example, the
17   analysis component 16 can provide an infrastructure map (i.e. a graphical
18   representation of the infrastructure as illustrated at 5). The collected information may
19   include not only the network configuration, but also software configuration
20   information, such as the version number as well as patch and bug-fix information of
21   installed software (e.g. operating systems, middleware and applications). Thus, the
22   analysis component 16 can monitor the software configuration status. It can also
23   analyze the collected data regarding the performance and health of the
24   infrastructure, and include these results in the graphical infrastructure representation.
25   Personnel from the customer's or the support-service provider's site can access
26   these results via an ISST web server 17 (which includes a web manager) and an
27   access service component 18. The analysis component 16 can also act as an "alarm
28   system" which detects imminent or already existing faults or malfunctions of the
29   infrastructure 5 and notifies the customer correspondingly via the web server 17.
30   Depending on the particular support service to be provided, the analysis component
31   may also initiate the steps which are necessary to remedy or avoid the fault or

1 malfunction, for example by sending corresponding instructions to the customer's
2 network administrator via the web server 17 or by arranging for corresponding
3 measures to be taken by external service personnel.

4 A further web server 19 is provided at the ISST 2 which allows HTTP access for
5 controlling and configuring the ISST 2.

6 In the preferred embodiment shown in Fig. 1, the topology data as well as the
7 collected data are stored at the customer's site, since commonly IP access to the
8 customer's infrastructure 5 from outside is restricted by a firewall (not shown).
9 However, in other embodiments (not shown) the discovery and collection data are
10 sent to the ISST 2 without being stored according to the data model at the customer's
11 site. In a further embodiment (not shown) only the information concerning the
12 topology is stored at the customer's site in order to allow a data collection with
13 reduced external access, but the collected information is sent to the ISST 2 without
14 being stored at the customer's site.

15 In contrast to the above, it is likewise possible to shift more "responsibility" from
16 the support-service provider's site to the customer's site. In particular, if the
17 bandwidth of the link 3, 4 between the IDT 1 and the ISST 2 is limited, it is
18 advantageous to reduce the amount of data to be transferred via this link. Therefore,
19 in a further embodiment a data consolidator 20 (shown with dashed lines in Fig. 1)
20 consolidates (e.g. filters) the collected data. Then, only the consolidated data are
21 sent to the ISST 2. In still further embodiments, the ISD 1 comprises an analysis
22 component (not shown), which already performs the entire topology and collection
23 data analysis or a part of it at the customer's site. Only data which represent the
24 results of the analysis are then sent to the support-service provider. The transferred
25 data may represent a fault message or an alarm which informs the ISST 2 that
26 measures to remedy or avoid the fault situation have to be taken.

27 Fig. 2 shows a more detailed architectural representation of a part of the IDT 1
28 of Fig. 1. The data collector 9 collects configuration and performance information of
29 the IT infrastructure 5 for example data concerning network interconnecting devices
30 (such as routers, switches) and software components (such as operating systems,
31 middleware and applications). The data collector 9 comprises a collection scheduler

1   21 and several sub-collectors for the different collection protocols: a DMI collector 22

2   and a SNMP collector 23 collect information about network devices, such as routers,

3   switches and hosts. A configuration file collector 24 collects data from configuration

4   files of devices. A WBEM collector 25 collects information about software

5   components.

6   The data collector 9 provides for the following functional options:

7   •   Collection on demand (immediate and synchronous, collection);

8   •   Collection according to a schedule on a regular (e.g. periodic) basis.

9   Particularly, it can run as a background task.

10   Interfaces provided by the collection configuration component 11 which can be

11   accessed e.g. by a user or infrastructure support manager via the Web server 13

12   (Fig. 1) are,

13   1.   as part of the specification of collection tasks:

14   •   the definition of what shall be collected (definition of a "collectable");

15   •   which device identification shall be used;

16   •   specification of schedules per collectible;

17   •   how or where to deliver the result;

18   2.   the initiation of a data-collection procedure (if the collection is to be carried

19   out on demand).

20   The following figures 3, 4 and 5 show the collector component 9 the collection

21   configuration component 11, and the transport office manager 10a together with the

22   IDT controller 6, in more detail. This is illustrated in Fig. 2 with dashed circles around

23   the corresponding elements.

24   Fig. 3 shows a more detailed functional architecture of the data collector 9. The

25   small circles depicted on the right-hand side of the small boxes indicate software

26   interfaces. The data collector 9 splits into three layers 31, 32, 33. The first layer, the

27   protocol layer 31, does the actual collection work. The components of this layer, the

28   DMI collector 22, the SNMP collector 23, the configuration file collector 24 and the

29   WBEM collector 24, use particular protocols (DMI, SNMP, WBEM) and access

30   methods (e.g. SNMP combined with TFTP) to collect information from infrastructure

31   elements. The second layer, the strategies layer 32, contains different collection

1   strategies, for example a sequential collection strategy 34 and a parallel collection

2   strategy 35. Sometimes it is preferable to use a mixed strategy rather than the pure

3   parallel strategy in order not to bother the infrastructure elements with too many

4   requests at a time. For example, with SNMP a restricted parallel strategy can be

5   used which prevents the collector from sending two SNMP requests at the same time

6   to a particular element. However, two or more requests can be sent to different

7   devices in parallel. The corresponding strategy is denoted with 36 in Fig. 3.

8   The third layer, the basic services layer 33 provides the basic functionality of

9   how to define a collection task. A collection scheduler 37 defines when a collection

10   has to take place. For example, a collection queue could determine that a collection

11   has to be carried out every full hour. A collectible schedule component 38 defines

12   what data have to be collected. A TFTP module 39 provides means to retrieve data

13   via TFTP.

14   Before any data can be collected, the following definitions of the collection task

15   have to be made:

16   - what to collect (with a collectible definition),

17   - when to collect (with a schedule),

18   - from which infrastructure element to collect (with element information),

19   - whom to notify when the collection is complete (with a notification object),

20   - where to deliver the result (with a result object), and

21   - who defined the task (with an identifier).

22   The dynamic behavior of the scheduler 9 is illustrated in Fig. 3 by reference

23   signs S1 to S9. In the first steps S1 and S2 the IDT controller 6 defines the following

24   items of the collection task: From which device to collect (IP address of the device),

25   what to collect (definition of collectible) and where to deliver the result (step S1) as

26   well as when to collect (schedule) (step S2). In the next step S3 the IDT controller 6

27   forwards the collection task to the collector on the protocol layer (here the SNMP

28   collector 23). Then in step, S4, the SNMP collector 23 configures the collection

29   scheduler 37 with the collection task. After step S4 the work is done for the collector

30   23. In step S5 the collection scheduler 37 fetches the collection schedule from the

31   collectible schedule component 38. The scheduler 37 holds a list of all scheduled

1 collection tasks. If the task is ready for collection (for example, when the point of time

2 when the collection shall be started has been reached) it passes the collection task

3 in step S6 to the strategy layer 32, in the example shown in Fig. 3 to the strategy 36

4 ("different elements in parallel"). The strategy 36 coordinates the access to the

5 infrastructure elements and initiates the actual collection (step S7). In step S7 the

6 protocol layer (here the SNMP protocol 23) retrieves the data about the infrastructure

7 and returns the result in step S8 to the IDT controller 6.

8   Fig. 4 shows a detailed functional architecture of the collection configuration

9 component 11 of Fig. 2. The collection configuration component 11 provides

10 information as to what data shall be collected for a given infrastructure element, what

11 is called a "collectible". The infrastructure elements are denoted as "devices" in Fig.

12 4. Collectible data may be from configuration files, log files, interface tables, routing

13 tables, health parameters, version, patch and update description data, usage, load

14 and performance data etc. The collectible definitions are contained in collectible

15 definition files, here an SNMP collectible definition file 43, a DMI collectible definition

16 file 44, a WBEM collectible definition file (not shown) etc. Device classes are listed in

17 a device-classes file 41. A relation file 42 contains relations between device classes

18 and collectibles. A collection configuration element 45 can retrieve a collectible

19 definition for a given device and a given collection protocol by first accessing the

20 device classes and relation files 41 and 42 in order to find out the correct collectible

21 for the given device, and then the SNMP collectible definition file 43 via a SNMP

22 configuration reader 46 (or the DMI collectible definition file 44 via a DMI

23 configuration reader 47, etc.). The files 41 to 44 are, for example, parsed with TCL

24 (Tool Control Language).

25   The sequence of a collection configuration task is indicated by reference signs

26 T1 and T2 in Fig. 4: In step T1, the IDT controller 6 requests a collectible for a given

27 device from the collection configuration component 11. The collection configuration

28 element 45 retrieves the collectible in the above-described manner and returns it to

29 the IDT controller 6. In step T2, the IDT controller 6 forwards the configuration task to

30 a protocol specific collector, here an SNMP data collector 48.

31   Fig. 5 shows a detailed functional architecture of the transport office manager

1   10a, which is implemented in the customer's IDT 1. A corresponding transport office

2   manager is implemented at the provider's ISST 2, which is indicated by 10b in Fig. 5.

3   The transport office managers 10a, 10b are operating-system independent. It is thus

4   possible to use different operating systems on both sides, for example Windows NT

5   on the one side and UNIX on the other side. The transport office managers 10a, 10b

6   are implemented independently of the transmission protocol and thus can be used

7   for, e.g., HTTP transmission or e-mail transmission. The transport office managers

8   10a, 10b have two functional parts, a send part and a receive part. Each transport

9   office manager 10a, 10b comprises a transport office manager element 51, which

10   controls the overall function, and a send module 52 and a receive module 53, which

11   are responsible for sending and receiving data. The sending of data from the IDT 1 to

12   the ISST 2 is performed according to the following sequence: In step U1, a file to be

13   sent is provided by the IDT controller 6. In step U2, the IDT controller 6 notifies the

14   transport office manager 10a that the file has to be sent, the manager 10a passes the

15   notification to the send module 52. The send module 52 fetches the file and forwards

16   it to an e-mail sender 54 or a HTTP sender 55 which performs the dispatch of the file.

17   The receipt of data from the infrastructure support-service tool 2 at the

18   infrastructure documentation tool 1 is performed according to the following sequence:

19   In step U3, the IDT controller 6 registers a call-back interface (depicted by a small

20   circle at step U4). In step U4, the transport office manager element 51 invokes the

21   registered interface. Then, in step U5 the receive module 53 fetches the received file

22   from an e-mail receiver 56 or an HTTP receiver 57 and forwards it to the IDT

23   controller 6 (or the ISST storage device, if the file is received at the support-service

24   provider's site).

25   Fig. 6 illustrates a hierarchy of different customer infrastructure elements that

26   can be subjected to the disclosed discovery and monitoring process. The hierarchy of

27   these elements forms what is called the distributed application stack of the

28   customer's IT infrastructure 5. It comprises the following elements in hierarchical

29   order: network infrastructure (such as routers and switches), storage system,

30   hardware (such as desktop computers and peripherals (such as printers), operating

31   systems, networking software, database software, middleware and utilities, software

1 applications.

2 The IT infrastructure 5 including these elements is mapped to a customer's

3 environment model, which also called a "data model", and is stored in the IDT

4 storage component 8 and/or the ISST storage component 14. An example of a

5 graphical representation of an instance of the data model is shown in Fig. 7. The

6 data model is object-oriented and uses an object-oriented database. An object of the

7 data model corresponds to each infrastructure element. Relations between

8 infrastructure elements (i. e. the topology of the infrastructure) are mapped to

9 corresponding relations between the objects, and features of the infrastructure

10 elements which are relevant for the disclosed monitoring process are modeled by

11 class attributes in the data model. The instance shown in Fig. 7 has a tree structure.

12 Depending on the actual IT infrastructures, other topologies, such as graphs, can be

13 modeled.

14 Examples of detailed data models of individual elements of an IT infrastructure

15 are shown in Figs. 8 to 10: Fig. 8 illustrates the data model of an element of the

16 lowest layer in the application stack, a network node (here: a router 71). A network

17 node is a hardware element which is link to a network, such as a server, a

18 workstation, a printer, a router, a switcher, a gateway, other interconnect devices,

19 etc. Fig. 9 illustrates the data model of another network node, a computer system 72.

20 Fig. 10 illustrates the data model of a storage system 73. The corresponding classes

21 are named "NetworkNode", "ComputerSystem" and "StorageSystem". The relation of

22 "StorageSystem" to "ComputerSystem" of Fig. 9 is included in classes "PhysicalDisk"

23 and "DiskController". A data model of a software application (not shown) can, for

24 example, include the installed version of the software application, patches, updates,

25 to the status of the application, its performance, etc.

26 Fig. 11 illustrates that the classes shown in Fig. 8 to 10 are inherited from other,

27 more general classes. In particular, as indicated by open arrows, "ComputerSystem"

28 and "StorageSystem" are generalized in to class "System", and "NetworkNode" is

29 generalized to a class "NodeElmt". In turn, these two classes are generalized to a

30 class "ExtensibleObject".

31 The mapping of the elements of the IT infrastructure 5 to classes is dynamical,

1   which is illustrated in Fig. 12. As most of the classes in the database schema (that is

2   the data model) are derived from ExtensibleObject 62, which in turn is derived from

3   PersistentObject 61, these classes can be dynamically extended at runtime by

4   creating and associating instances of any of the subclasses of Property 63. In

5   particular the preferred embodiment provides for ScalarProperty 64, ArrayProperty

6   66 and GroupProperty 65. Each ScalarProperty object can store all kinds of scalar

7   values, such as integers with varying precision, floating point numbers with varying

8   precision, strings of virtually arbitrary length or binary data of virtually arbitrary

9   length. Alternatively a ScalarProperty object can also contain a reference to a

10   different object in the same or a different database. This mechanism is also used to

11   dynamically store associations between objects in the database. An ArrayProperty

12   object consists of ScalarProperties objects, which can be accessed using an index.

13   An ArrayProperty is dynamic not only in the size, but also in the number of

14   dimensions it has. For example, in the one dimensional case it represents a vector, in

15   the two dimension case it is a table. A GroupProperty object provides for grouping

16   other Property objects. This means can be used for grouping Properties, for example,

17   information related to a particular network protocol, e.g. UDP, can be put into a single

18   GroupProperty. The information on a different protocol, e.g. TCP can then be put in a

19   second group. As GroupProperty is derived from Property, and as a GroupProperty

20   object groups other Property objects, this also means that a GroupProperty object

21   can contain other GroupProperty objects. In other words, GroupProperty objects can

22   be nested.

23   The dynamic extensibility of the databases schema is achieved by the use of

24   meta classes. Such meta classes have attributes which are classes. This is

25   illustrated in the table of Fig. 13: The left-hand column shows the logical level. The

26   lowest level is the Object level, the medium level is the Class level, and the highest

27   level is the Meta level. Commonly, only the Object level and the Class level are used

28   in object-oriented programming. The column in the middle indicates the name of the

29   "concept" used in these levels, namely "Object", "Class" and "MetaClass". The right-

30   hand column shows a concrete example (an instance) of each of these concepts. On

31   the Object level, the shown object is of the type "laserprinter" of the vendor "Hewlett-

1 packard". On the Class level, the shown object is a class "printer" with the attributes

2 "type" and "vendor". On the Meta level, the shown object is "MetaClass" which is a

3 class of classes. Its (meta) attributes are "classname" (here: "printer") and

4 "attributes" (here "type", "vendor"). The implementation of such a meta level allows

5 the instantiation of new classes during runtime, without a change of the database

6 schema.

7   A preferred embodiment of a remote support-service method carried out with

8 the preferred embodiments of the remote support-service system of Fig. 1 is

9 illustrated in Fig. 14. Steps V1 to V10 are carried out by the IDT 1 at the customer's

10 site, whereas steps V11 to V13 are carried out by the ISST 2 at the support-service

11 provider's site. The method is carried out automatically and periodically as a

12 background task, but can also be executed on demand by the customer or support-

13 service provider.

14   In step V1, the discovery component 7 performs the discovery task. It discovers

15 changes in the IT infrastructure 5, for example, the appearance of a new

16 infrastructure element or the modification or disappearance of an already existing

17 infrastructure element, including the inter-relation between infrastructure elements (i.

18 e. the infrastructure topology). In step V2, it is ascertained whether a change in the IT

19 infrastructure 5 has been discovered. If the answer is positive, the discovery

20 component 7 informs the IDT controller 6 correspondingly. In step V3, the IDT

21 controller 6 finds out, if a new infrastructure element has been discovered, what that

22 element is and what data have to be collected for it, by consulting the collection

23 configuration component 11. In steps V4 and V5, the IDT controller 6 initiates the

24 mapping of the IT infrastructure into a corresponding data model in the storage

25 component 8, and configures the data collector 9 correspondingly. If the answer in

26 step V2 is negative, the process does not carry out steps V3 to V5, but continues

27 with step V6. In step V6, the collector 9 collects the data to be collected from the

28 infrastructure 5 and returns them to the IDT controller 6. In step V7, the IDT controller

29 6 inputs the collected data into the storage component 8. In step V8, the IDT

30 controller 6 decides whether the data are to be transferred to the ISST 2. If the

31 answer is negative, the flow returns to step V1. If the answer is positive, in step V9

1   the IDT controller 6 prepares a data file to be transferred. In some embodiments, the

2   data preparation step V9 includes a consolidation of the data to be transferred. In

3   step V10, the IDT controller 6 instructs the transport office manager 10a to send the

4   prepared file to the ISST 2. The steps V1 to V8 are carried out in a permanent loop,

5   depending on a collection schedule in step V6 (for example, every full hour). The

6   decision that data are to be transferred to the ISST in step V8 can be taken

7   depending on the result of the collection in step V6. For example, the data transfer

8   may periodically be carried out at greater time intervals than the collection period, if

9   no (or no relevant) change has been found in the discovery and data collection, but

10   is carried out immediately if such a change has been detected.

11   The remaining steps V11 to V13 are performed out at the support-service

12   provider's site. In step V11, the file sent from the customer's site is received. In step

13   V12, the file is stored in the ISST storage component 14 and analyzed by the

14   analysis component 16. In step V13, actions are taken depending on the results of

15   the analysis. A standard action is, for example the provision of a status report. If

16   faults, malfunctions, outdated software versions etc. have been detected, the

17   corresponding action may be the triggering of an alarm, the instructing of service

18   personnel, the triggering of a software update action etc.

19   Fig. 15 illustrates an embodiment wherein the support service is provided by

20   several co-operating sub-services which may be located at different sites, namely a

21   service and support portal 2a, several problem-domain-specific diagnostic services

22   2b to 2e and an overall support provider 2f. The customer communicates with the

23   services and support portal 2a in order to subscribe to a support service. The

24   particular service can be individually tailored to the particular customer's IT

25   infrastructure 5 and the customer's specific needs (for example, his specific security

26   requirements).

27   The problem-domain-specific diagnostic services 2b to 2e are specialized to

28   provide support for specific parts of the customer's IT infrastructure 5. They are, for

29   example, a NT support-service tool 2b, an UNIX support-service tool 2c, a network

30   support-service tool 2d and generalized diagnosis support-service tool 2e. The

31   customer's IDT 1 is equipped with a data distribution component (not shown) which

1   knows what data are relevant for which one of the problem domains specific

2   diagnostic services 2b to 2e, and groups and addresses the data to be transferred to

3   the corresponding one of the services 2b to 2e.

4   In addition, the IDT 1 keeps the overall support provider 2f informed about any

5   data transfer to the services 2a to 2e by transferring corresponding notifications to it.

6   The results obtained by the data analysis carried out by the problem-domain-specific

7   diagnostic services 2b to 2e are transferred directly to the overall support provider 2f.

8   The overall support provider 2f collects the results from the services 2b to 2e,

9   sends overall reports and alarms to the customer 1 (which are called "Trouble

10   Tickets" in Fig. 15) and provides an overall monitoring facility for the customer 1 and

11   the co-operating sub-services 2a to 2f via IP links. The messaging can be based on

12   XML.

13   Fig. 16 describes steps S1 to S5 of Fig. 3 in detail. The collection is a two-step

14   process. In the first step, a client application (i.e. the NDT controller 6) configures a

15   collection task and passes the task to the collector. Fig. 6 shows what happens when

16   a new collectible is inserted by the client, i.e. the procedural steps during insertion of

17   a new task.

18   When the client application adds a new task to the collector, the collector

19   checks whether the collection task is valid. In this embodiment, validity means that

20   the task has configured the following attributes:

21   •   A schedule;

22   •   A valid collectible definition;

23   •   Device information that contains the IP address and other access

24   parameters for the collectible;

25   •   A non-null session identification that defines the application that defined

26   the task.

27   If the task is not valid, it is rejected with an error message. The next step is to

28   check whether the collectible definition matches the collector, e.g. that the SNMP

29   collector gets only SNMP collectible definitions and not DMI collectibles. The task is

30   rejected when the collectible does not match the collector. In the positive case the

31   collector forwards the task to its scheduler. The scheduler determines the date and

1   time of the first collection and inserts the task into its queue.

2   Fig. 17 describes steps S6 to S8 of Fig. 3 in detail. The scheduler has an

3   internal priority queue that holds a list of all collection tasks sorted by time. When a

4   collection task is ready, the steps shown in the flowchart depicted in Fig. 17a, which

5   is the first part of an entire flow chart continued in Fig. 17b, are executed. It is

6   noteworthy that the tasks are executed for every collection task that has to be

7   performed. Fig. 17 particularly shows an exemplary scheduling (a) and applying

8   strategy (b) by means of a task execution process flowchart.

9   At the beginning, the scheduler removes the task from the priority queue and

10   determines the next collection time. Sometimes there will be no next collection time,

11   e.g. in the case of a collect once schedule. If there is a collection time, the scheduler

12   will insert the task with the new collection time again. Otherwise the task will not be

13   inserted into the queue and therefore not handled again (i.e. collect once).

14   The next step checks whether the task should be forwarded to a corresponding

15   strategy. If the task was suspended due to repeated errors, the scheduler will check

16   whether to restart the task again. If it should be disabled, the task is finished.

17   Otherwise the scheduler will change the status of the task to 'active' and pass the

18   task to the strategy. If the task was not suspended due to errors, it may be

19   suspended at this point. Otherwise the task will be forwarded to the corresponding

20   strategy.

21   The strategy holds a list of all collection tasks that have to be performed as fast

22   as possible and passes the tasks, in accordance with the strategy, to the respective

23   collection method. The collection method tries to retrieve the collectible. If the

24   collection succeeds, a retry counter is reset which is used for suspending tasks that

25   resulted in several repeated errors. Further it delivers the result and notifies it to the

26   client if applicable. If the collection fails, the strategy increments the retry counter.

27   The task is when the counter reaches a maximum.

28   Figs. 18 and 19 a-c show interfaces and their hierarchies. Fig. 18 shows

29   interfaces for collectible definitions. Common to all collectible definitions is a 'name'

30   and a 'unique identifier'. Protocol specific information is provided by derived protocol

31   specific interfaces. For instance, the interfaces 'ISNMPCollectibleDef' define items

1   that can be retrieved via SNMP.

2   Referring now to Fig. 19a, strategies are used to control access to a device.

3   The base interface 'IcollectionStrategy' consists of two methods. The method

4   'CollectionMethod' sets the collection method that is used in conjunction with that

5   strategy. The collection method is preferably part of the protocol. The interface

6   'IparallelCollectionStrategy' is an inherited interface from 'IcollectionStrategy'. It has

7   additional methods to set the maximum number of threads and to retrieve the number

8   of currently active threads.

9   Now referring to Fig. 19b, the first group of interfaces defines collection

10   schedules. The collection schedules are used by the scheduler in order to find out

11   when to perform a collection task. The interface basically provides two methods. One

12   returns the date for the 'first' collection and the second method returns the date of the

13   'next' collection.

14   Referring to Fig. 19c, a family of device information interfaces define how to

15   access a device. The basic interface contains only the network address of the

16   device. Protocol specific information like SNMP community strings, retry and time-

17   outs are defined in protocol dependent interfaces. It is noted that the device

18   information is protocol dependent. For example, an SNMP collection task needs the

19   SNMP community strings. These strings are not provided by the base interface

20   'IDeviceInfo'. In order to define SNMP collection tasks, the interface

21   'ISNMPDeviceInfo' is to be used. This interface is derived from the interface

22   'IDeviceInfo' and extended by commonly known methods (algorithms) to retrieve and

23   set the community strings.

24   Finally, Fig. 20 shows the DMI collector 22 of Figs. 2 and 3 and a

25   corresponding algorithm in greater detail. The DMI collector retrieves arbitrary DMI

26   groups and DMI tables. A DMI collectible is defined by the above described interface

27   IDMICollectibleDef'. A DMI collectible is particularly defined by the following

28   attributes:

29   •   The component name;

30   •   The class name for the DMI group or table;

31   •   A list of IDs.

1   The DMI collector performs the following steps for each collectible:

2   • Enumerate all components for a device

3   • For each component that matches the component name in the collectible

4 definition:

5   • Enumerate all DMI classes in the component;

6   • For each class that matches the class name in the collectible definition:

7   • Collect the item and return the result.

8   Thus, a general purpose of the disclosed embodiments is to provide an

9 improved system, computer program product and a method for providing a remote

10 support service.

11   All publications and existing systems mentioned in this specification are

12 herein incorporated by reference.

13   Although certain systems, methods and products constructed in

14 accordance with the teachings of the invention have been described herein, the

15 scope of coverage of this patent is not limited thereto. On the contrary, this

16 patent covers all embodiments of the teachings of the invention fairly falling

17 within the scope of the appended claims either literally or under the doctrine of

18 equivalents.